The Analytical Community and the High Level Architecture A Happy Marriage

John W. Ogren
The MITRE Corporation
11493 Sunset Hills Dr.
Reston, Va. 22090
jogren@mitre.org

KEYWORDS
Analysis, HLA, Time Management, FOM, SOM, Eagle, JTFp

ABSTRACT
The High Level Architecture (HLA) has presented the simulation community with a unique opportunity to leverage the power of cooperating simulations across a broad range of applications. The HLA provides the framework and tools to accomplish this interoperability. Many challenges still exist and the HLA is not a magical solution. Yet the HLA over the past year has shown that it can flexibly accommodate and support the interoperability of simulations ranging from training to analysis. Each of the broad categories of application has its own special needs and places its own unique requirements on the use of the HLA, more specifically on the Run Time infrastructure (RTI). This paper will describe the involvement of the Analytical community in the definition and development of the HLA. One of the applications used to prototype and frame the analytical community's concern was the Eagle combat model owned and maintained by the TRADOC Analysis Center (TRAC). This paper presents an overview of the technical approach taken to incorporate the RTI into the legacy Eagle model code, the development of the Eagle Software and Federation Object Models, the time management scheme used to ensure causality and consistency among the federates and a summary of the lesson learned and results.

1.0 INTRODUCTION

To validate the needs of the analytical community in the development of the HLA, the Eagle model was used in two applications (called federations) which tested the HLA's ability to support a distributed analytical model (the Eagle Early Analysis Experiment) and to support the sharing of functionality between service unique (Army, Air Force, and Navy) models (the JSIMS Protofederation).

2.0 THE EAGLE DESIGN

Eagle was developed in the late 80's as a vehicle to investigate the application of artificial intelligence to explicitly model command and control in a combat simulation. The model's typical combat functionality (such as attrition adjudication) relies on the extensive combat modeling experience at TRAC and is rooted in standard, validated algorithms. The model is categorized as a constructive, aggregate Corps level model with normal resolution to company size units. Eagle uses a hybrid event structure that relies on both the notion of continuous time using time steps (1 to 5 min) and the projecting of discrete events limited to the duration of the time step. Eagle's architecture is built on the object-oriented programming paradigm. This paradigm is based on a philosophic focus on data grouped into objects acting on other objects, rather than on the traditional focus on processes which act on data. The data or knowledge representation reflects the unique requirements of the military domain. Eagle's knowledge representation conforms to the user's understanding of the problem space in three main areas. First, military units, weapon systems, and munitions are defined as objects. Second, terrain is represented as a network of mobility corridors each of which are objects. Third, plans are represented in standard five-paragraph field order format, so that the user can specify orders to units in a mission-oriented manner. Simulated command posts respond to these orders by accessing its domain knowledge executing the mission as directed. Decisions are made by the software commander and the information or directives are passed up and down the chain of command. This flow of information between command posts (software objects) is very important, because the actions of the units are not scripted based on time, but occur based on events that cause commanders to give their approval to execute the next portion of the plan. A software commander's perception of the battlefield is based solely on what his subordinates and the intelligence system are telling him and how it relates to the command's battle plan. Eagle portrays ground maneuver, attack helicopter, field artillery, air defense, air and ground intelligence units or assets and engineer units. The primary emphasis is on Army units and capabilities, yet Eagle also plays air force air assets used in support of ground operations.

2.0 EAGLE ARCHITECTURE

The Eagle architecture is displayed in the figure 1. The highest abstraction of this architecture consists of three entities: The Application Entity which in the figure consist of the Eagle applications and the Eagle framework, the Application Platform Entity, and the External Environment Entity.

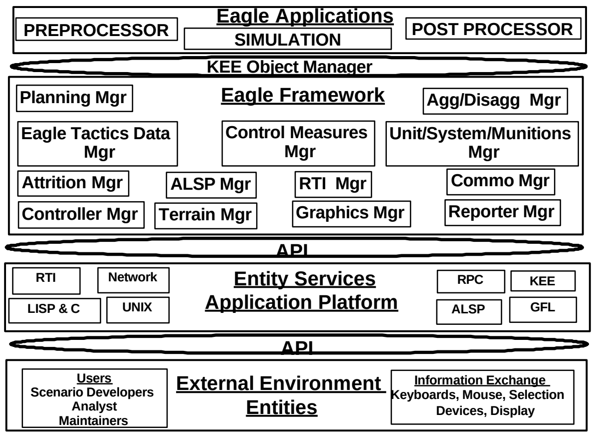


Figure 1 Eagle Architecture

Combat units exist in the simulation portion of the Eagle Applications. The actions of these units are modeled by the Eagle Framework Services. These services maintain ground truth information and provide requested information or services to the combat units playing in the simulation. For example: If a unit desires a route, it will query the Terrain manager. Likewise, if the unit desires to send a message to another unit, it will send its message to the communications manager which will direct the message to the recipient and apply necessary delays to its arrival. The remaining entities, the platform services and external environment provide the software and hardware upon which the Eagle model operates. The Run Time Infrastructure exits within both the application services and as a portion of the external environment for it provides the communications necessary for our distributed computing capability.

3.0 SUMMARY OF EFFORT

Eagle participated in two federations, the Eagle Early Analysis Experiment and the Joint Training Federation. The relationship and general configuration of the two efforts are shown in Figure 2.

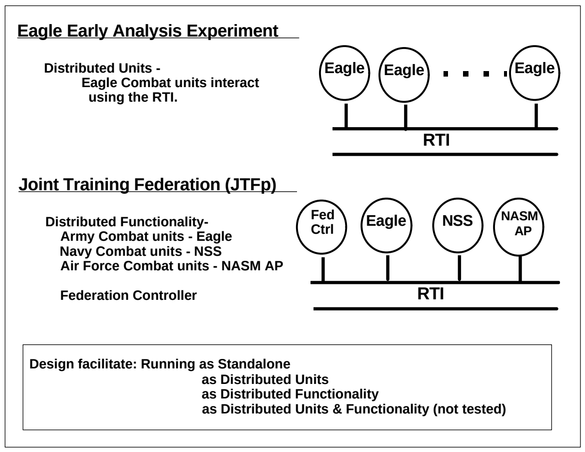

Eagle Early Analysis Experiment
Distributed Units - Eagle Combat units interact using the RTI.
Eagle
Eagle
Eagle
RTI
Joint Training Federation (JTFp)
Distributed Functionality-
Army Combat units - Eagle
Navy Combat units - NSS
Air Force Combat units - NASM AP
Federation Controller
Fed Ctrl
Eagle
NSS
NASM AP
RTI
Design facilitate: Running as Standalone
as Distributed Units
as Distributed Functionality
as Distributed Units & Functionality (not tested)

Figure 2 Summary of Effort

3.1 Eagle Early Analysis Experiment

The goal of the Eagle Early Analysis Experiment was to modify the distributed version of Eagle to use the communications backbone of the High Level Architecture's Runtime Interface. Eagle had previously been distributed using the Aggregate Level Simulation Protocol (ALSP). The original goal of distributed Eagle was to develop a distributed architecture in such a way that Eagle can maintain the same temporal, tactical, organizational and spatial consistency on multiple processors that currently exists on a single processor. The purpose was to leverage a distributed environment using multiple processors on a LAN to increase the execution speed of the Eagle model.

The distributed Eagle design distributes the combat units among multiple Eagle simulations on the network. The common Eagle software environment is duplicated on each machine or federate on the network and combat units are selected to run on one of the federates. Each Eagle federate maintains its own set of core services such as terrain, attrition, tactics knowledge base, and control measures data. The distributed design uses interactions to maintain consistency between these services. For example, if a minefield is laid on one federates terrain data base, then that minefield must be created in the data bases of each federate. The distributed design accommodates any mix or match of combat units on a federate. However, a combat unit will only exist as an actual unit in one federate. All other federates maintain a reflected representation of the unit. The distributed design maintains consistency between the actual combat units and their reflected representations.

3.2 JSIMS ProtoFederation

The goal of Eagle as a member of the JSIMS ProtoFederation was to allow externally generated objects (combat or environmental) to interact with Eagle generated combat units while maintaining the same or better temporal, tactical, organizational and spatial consistency that currently exists within the Eagle model. The purpose was to replace Eagle's air operations functionality with that of the other federation members. Air Force assets are played by the National Air and Space (Warfare) Model Advance Prototype (NASM-AP) and Naval Air assets are played by the Naval Simulation System (NSS). Each of these simulations is a service sponsored simulation meeting their standards of validation. Thus, by Eagle joining this federation it receives validated, verified modeling algorithms approved by each service and therefore the representation of air operations is better. Eagle provides all ground combat functionality for the federation. All ground units are reflected in the member federates. Eagle replaces its normal fixed wing operations by subscribing to the federation air objects and by publishing and subscribing to interactions between the ground and air players.

The software design allows Eagle to be run as a standalone model, or as a member of each federation individually or as a combination.

4.0 TECHNICAL APPROACH

The technical approach taken to incorporate the RTI into the legacy Eagle code was focused on leveraging Eagle's object oriented implementation and its software architectural design to seamlessly incorporate the notion of external combat objects and events acting on the Eagle federate combat units. A main consideration when incorporating the RTI was that no additional overhead could be incurred by the standalone version of Eagle because of its distributed capability. The standalone model must run the same and execute at its normal speed.

4.1 Technical approach within Eagle

Within Eagle, the majority of the effort was to modify the Eagle Services which exist in the Eagle Framework (figure 1). When the Eagle model is executing as part of a federation, these Services inherit new functionality, or in the case of the Eagle controller, new events. This gives the Services the ability to recognize the need to interface appropriately with the RTI. Figure 4 depicts a typical modification to one of the Eagle services. Within Eagle all interactions between units go through a Service. These Services are responsible for modeling the interactions. In the example shown in Figure 4, when a combat unit sends a communications message to another unit, the message goes through the communications manager. The communications manager will normally just model the delays that would be expected due to type of equipment, jamming or the combat state of the units communicating and deliver the message to the receiving objects communications interface. However, when Eagle is interacting with the RTI, the communications manager has the additional functionality to determine if the receiving unit exists

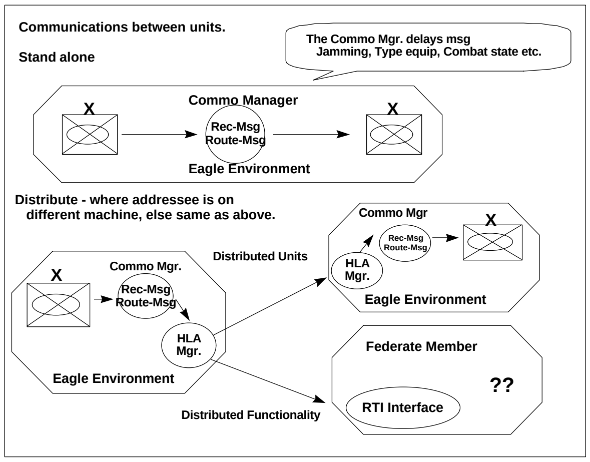


Figure 4 Technical Approach Internal Eagle

on the local federate. If it does not and is only a reflected representation, the communications manager has the additional responsibility to place the message in an interaction and deliver it to the HLA manager (a new Eagle Service) for publishing on the RTI. Eagle federates that have subscribed to this interaction will receive the message. The HLA manager on the receiving federate will determine if the addressee is real (not reflected); and if it is, will deliver the message to the local communications manager just as it would be received in the standalone version. The local communications manager treats this message as any other and determines that the addressee is actually played on the federate and deliver s the message.

This diverting of interactions to the RTI is duplicated within each Eagle Service and provides the seamless interface with the RTI. The Eagle user, when desiring to set up a federation or join an existing federation (such as JTFp), must only set an Eagle load parameter that causes the Eagle initialization code to establish the connection to the RTI and insures that the correct functionality has been inherited to the Services.

4.2 Technical approach between Eagle and RTI

The current version of the RTI requires that its runtime code be incorporated as a normal part of the federate. Therefore, a basic requirement of using the RTI is that the federate must be executing the same type of code as the RTI. Currently the RTI only supports models that are written in C++. Eagle is written in LISP. Therefore as any other model that is not written in C++ (such as models using ADA), a separate interface must be created to act as a broker between the model's code and the RTI's code. Figure 5 depicts the approach taken by Eagle to provide this interface to the RTI.

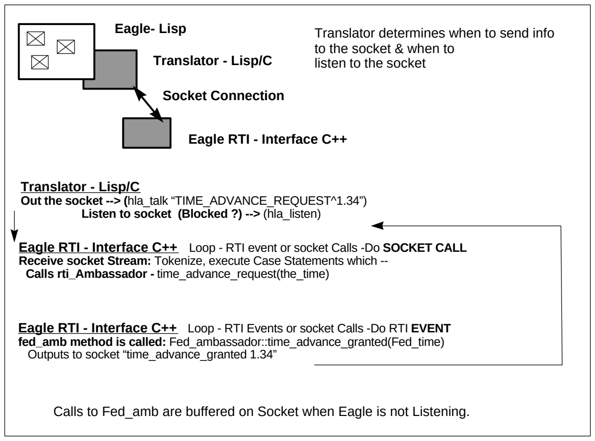

Eagle- Lisp
Translator - Lisp/C
Translator determines when to send info to the socket & when to listen to the socket
Socket Connection
Eagle RTI - Interface C++
Translator - Lisp/C
Out the socket --> (hla_talk "TIME_ADVANCE_REQUEST^1.34")
Listen to socket (Blocked ?) --> (hla_listen)
Eagle RTI - Interface C++ Loop - RTI event or socket Calls -Do SOCKET CALL
Receive socket Stream: Tokenize, execute Case Statements which --
Calls rti_Ambassador - time_advance_request(the_time)
Eagle RTI - Interface C++ Loop - RTI Events or socket Calls -Do RTI EVENT
fed_amb method is called: Fed_ambassador::time_advance_granted(Fed_time)
Outputs to socket "time_advance_granted 1.34"
Calls to Fed_amb are buffered on Socket when Eagle is not Listening.

Figure 5 -- Technical Approach - External to Eagle

For Eagle's interface to the RTI, a separate executable was created written in C++ that brokers the functionality between the RTI and Eagle. The interface is called the Eagle Common Module (ECM), which is a separate process running on the same machine as it's attached federate. Within Eagle the Lisp code that includes the HLA Manager (see Technical approach within Eagle) plus the additional code to establish the communications with the ECM is included in the HLA Translator. Each Eagle federate's HLA manager communicates with the RTI through a socket connection to the ECM. The ECM provides the interface with the RTI's declared Federate and RTI ambassador C++ objects that have the functionality as specified in the HLA RTI Interface Specification. A typical interaction between the Eagle HLA manager, the ECM and the RTI is shown in figure 5. The HLA manager in response to a request from the Eagle controller service requests to advance time. To accomplish this, it forms a message and sends it to the ECM via the socket connection and then begins listening to the socket for a return answer. The ECM C++ "main" is basically a continuous loop that is either waiting for a call from the HLA manager or allowing the RTI processing time. In this case, the ECM receives the message, determines the type, and calls the appropriate RTI ambassador service with the provided parameters: timeAdvanceRequest (fed_time). The ECM then loops allowing the RTI processing time which in turn allows the RTI to call ECM declared Federation ambassador services. In this case when the RTI determines that the requesting federate can advance to the new federation time, it calls the declared Federation ambassador service timeAdvanceGranted. This service receives the call and forms a message to output back to the HLA manager again via the socket connection. The HLA manager which is waiting for the answer receives the message, determines the type, and passes the answer back to the controller. Calls to the ECM from the RTI (through the Federate ambassador) are buffered on the socket when the HLA manager in Eagle is not listening to the socket.

This approach of providing a separate process (the ECM) within the Eagle framework to service the RTI has proven to be an acceptable alternative to the preferred method of including the RTI within the normal simulation code. The ECM supports all the interface specification. However, additional functionality must be included in the ECM as this interface matures to handle federation ambassador calls that require an immediate return answer, such as requests for transfer of attributes. An approach that opens an additional socket connection to a separate Eagle Lisp process has been investigated and will be incorporated as needed.

5.0 SOFTWARE & FEDERATION OBJECT MODELS

As an integral part of the HLA process, simulation models must define the information that they are willing to share, called the Software Object Model (SOM), and the information that they are actually sharing as part of a unique federation, called the Federation Object Model (FOM).

The actual class structure in the model (if it exists) is not necessarily the class structure used in the simulation's SOM. For example, a typical command unit in Eagle, such as a blue brigade command post (BLUE_BDE_CP), would inherit or have available assets, attributes, and physical and cognitive functionality from 66 separate classes (figure 6). However, much of this class structure is not needed

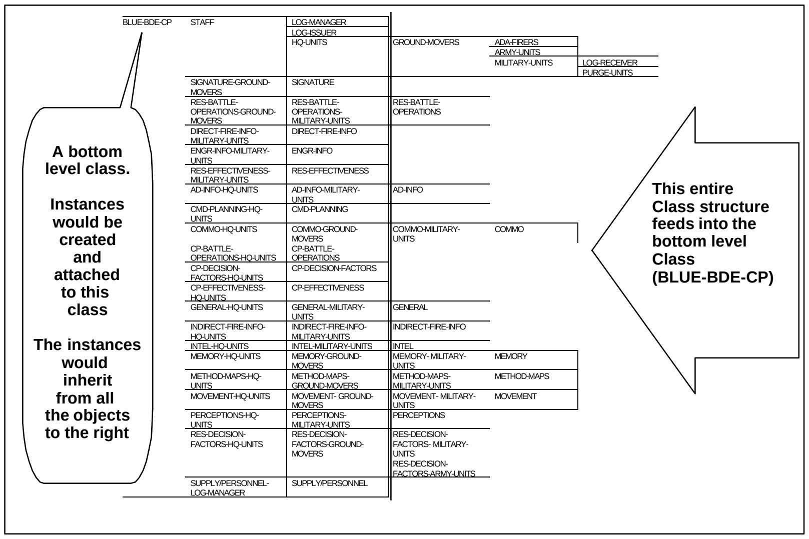


Figure 6 Normal Eagle Class Strucuture

when determining the minimum necessary information for the class structure defined in the SOM (See Figure 7). For Eagle, only those classes that had separate information to be published were included in the SOM. In fact, even though the classes in the SOM are actual classes in the Eagle Class Structure, the information associated with them may come from a separate class. For example, the attribute "percent effectiveness" comes from the Eagle object CP-Effectiveness, but for the SOM it is included as an attribute associated with HQ-UNITS. The internal complexity of how Eagle manages data is hidden from the user. The goal is to educate the user not overwhelm him.

Figure 7 depicts a portion of the Eagle SOM class structure and the associated FOM class structures for the Eagle Early Analysis Experiment and the JTFp. Of interest is the translation from the Eagle SOM to the JTFp FOM. Note that the Eagle internal class structure had to be translated to a class name that was a

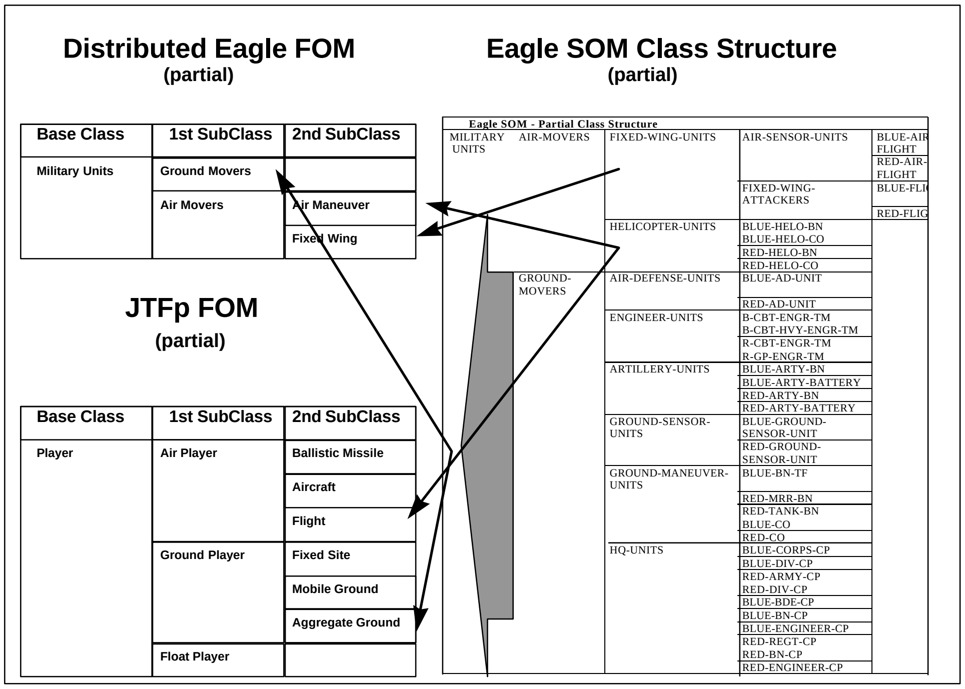
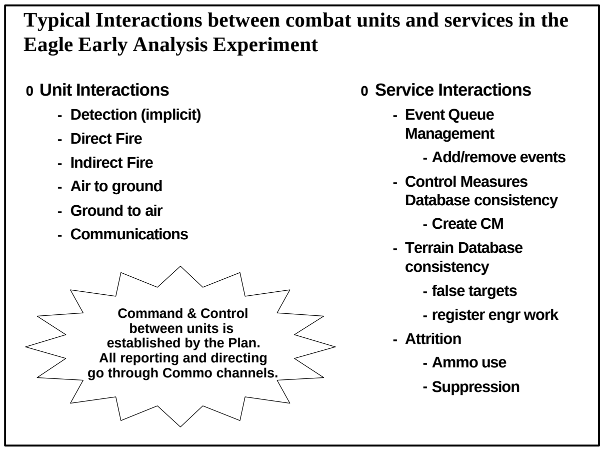


Figure 7 Eagle SOM and FOM Example

compromise among the federates. This is an example of the negotiations that must take place between the federates when compiling a FOM.

The class structure that is defined in the FOM must be of some practical use. Therefore attributes are associated with each class. The types of attributes that are defined are up to the federates in negotiation with the federation. However in the Eagle Early Analysis Experiment and the JTFp, the driving requirement for the number and type of attributes was that information needed by the reflecting federate to drive its detection algorithms (Figure 8). Within a typical Eagle combat unit, it requires approximately 450 attributes for a unit to maintain its perceived state of itself and subordinates. Of these 450 attributes, only 43 are need to be published for ground maneuver units to drive the Eagle ground detection process, and only 29 need to be published to drive the detection processes on the member federates of the JTFp federation. Again translations are required from Eagle's normal nomenclature to that used by the JTFp. For example, Eagle characterizes a unit by "side," whereas in the JTPp the same attribute is called affiliation.

In addition to the object class structures and attributes, interactions must be defined to affect the actions between players in the simulations that are not played on the same federate. With the Eagle Early Analysis Experiment, the interactions defined can be categorized in two groups: those interactions that affected actions between combat players and interactions that maintained the consistency between the services on the separate Eagle federates (Figure 9).

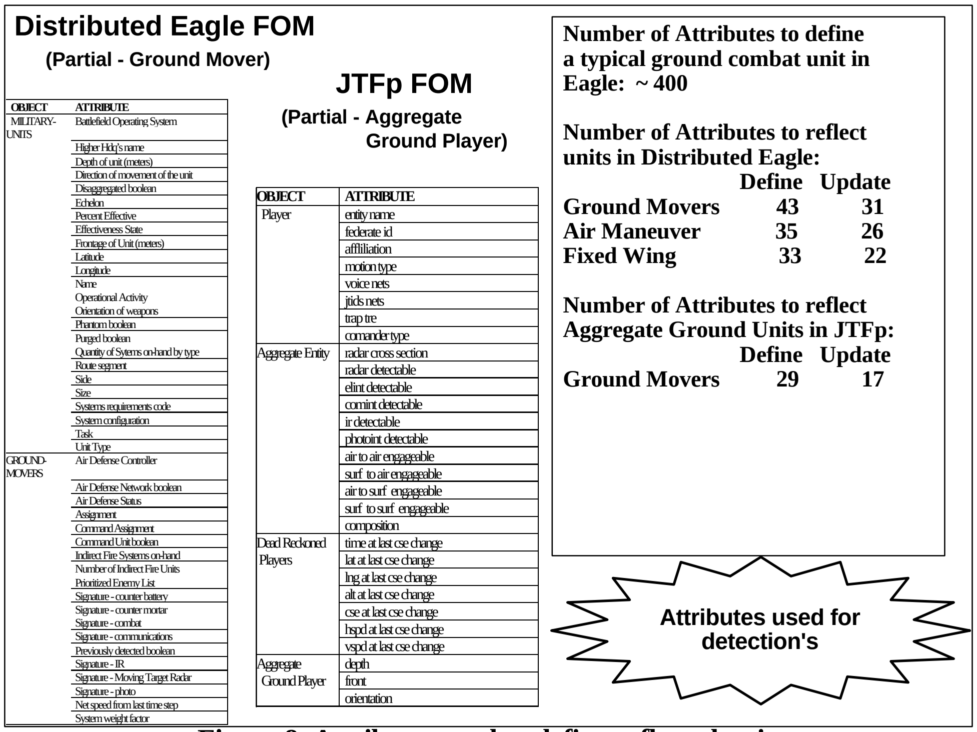

Typical Interactions between combat units and services in the Eagle Early Analysis Experiment

- Unit Interactions
  - Detection (implicit)
  - Direct Fire
  - Indirect Fire
  - Air to ground
  - Ground to air
  - Communications
- Service Interactions
  - Event Queue Management
    - Add/remove events
  - Control Measures Database consistency
    - Create CM
  - Terrain Database consistency
    - false targets
    - register engr work
  - Attrition
    - Ammo use
    - Suppression

Command & Control between units is established by the Plan. All reporting and directing go through Commo channels.

Figure 9   Eagle Early Analysis Interactions

As indicated previously in the Eagle distributed design, the choice was made to distribute units not services. The services are duplicated on each Eagle federate. Therefore it is incumbent upon the HLA manager on each federate to insure that as local changes are made to its knowledge bases, that that information is sent out to all federates in the federation. These interactions are referred to as data base consistency interactions. For example, if attrition occurs within a federate and


Distributed Eagle FOM

(Partial - Ground Mover)

OBJECT	ATTRIBUTE
MILITARY-UNITS	Battlefield Operating System
	Higher Hdq's name
	Depth of unit (meters)
	Direction of movement of the unit
	Disaggregated boolean
	Echelon
	Percent Effective
	Effectiveness State
	Frontage of Unit (meters)
	Latitude
	Longitude
	Name
	Operational Activity
	Orientation of weapons
	Phantom boolean
	Purged boolean
	Quantity of Sytems on-hand by type
	Route segment
	Side
	Size
	Systems requirements code
	System configuration
	Task
	Unit Type
GROUND-MOVERS	Air Defense Controller
	Air Defense Network boolean
	Air Defense Status
	Assignment
	Command Assignment
	Command Unit boolean
	Indirect Fire Systems on-hand
	Number of Indirect Fire Units
	Prioritized Enemy List
	Signature - counter battery
	Signature - counter mortar
	Signature - combat
	Signature - communications
	Previously detected boolean
	Signature - IR
	Signature - Moving Target Radar
	Signature - photo
	Net speed from last time step
	System weight factor

JTFp FOM

(Partial - Aggregate Ground Player)

OBJECT	ATTRIBUTE
Player	entity name
	federate id
	affliliation
	motion type
	voice nets
	jtids nets
	trap tre
	comander type
Aggregate Entity	radar cross section
	radar detectable
	elint detectable
	comint detectable
	ir detectable
	photoint detectable
	air to air engageable
	surf to air engageable
	air to surf engageable
	surf to surf engageable
	composition
Dead Reckoned Players	time at last cse change
	lat at last cse change
	lng at last cse change
	alt at last cse change
	cse at last cse change
	hspd at last cse change
	vspd at last cse change
Aggregate Ground Player	depth
	front
	orientation

Number of Attributes to define a typical ground combat unit in Eagle: ~ 400

Number of Attributes to reflect units in Distributed Eagle:

	Define	Update
Ground Movers	43	31
Air Maneuver	35	26
Fixed Wing	33	22

Number of Attributes to reflect Aggregate Ground Units in JTFp:

	Define	Update
Ground Movers	29	17

Attributes used for detection's

Figure 8 Attributes used to define reflected units

possible false targets are created and saved in the terrain data base, then the number of false targets and their locations must be conveyed to the other federates. Interactions are used to deliver this information. Distributed Eagle made no assumption as to which units have to play on a particular federate; therefore, all possible interactions that could normally occur between units in a standalone version must be defined in the FOM. A total of 23 interactions are defined to satisfy both the actions between units and services. A typical sequence of interactions that may occur in Eagle is shown in Figure 10. In one Eagle federate, a mechanized infantry company and a field artillery battalion are played. In the second Eagle federate, the Brigade headquarters that is controlling the infantry and artillery units and an enemy company is played. The sequence of interactions begins with the infantry company that requests a call for fire. Of the 11 events that occur, six result in RTI interactions between units because they do not exist on the same federate.

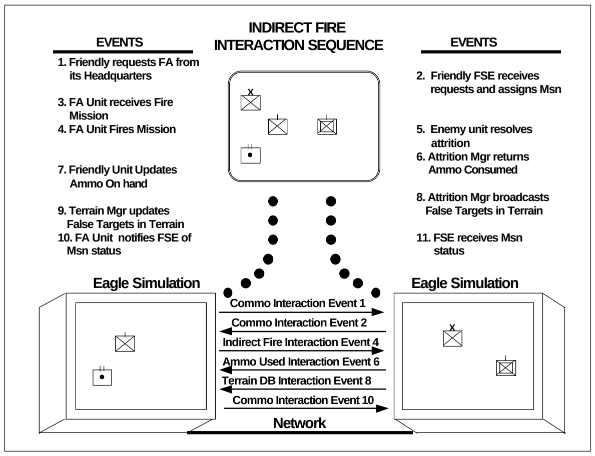
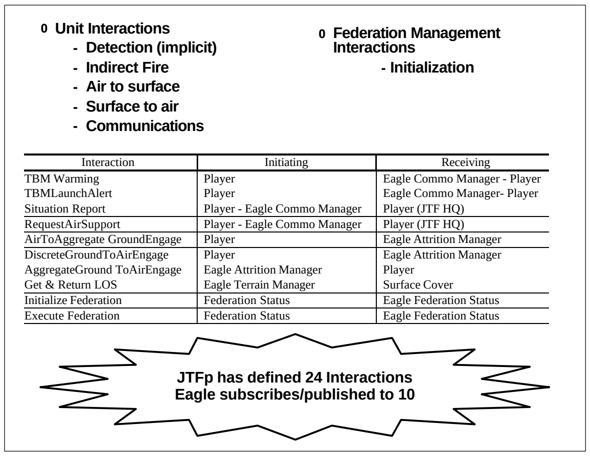

INDIRECT FIRE INTERACTION SEQUENCE
EVENTS
1. Friendly requests FA from its Headquarters
3. FA Unit receives Fire Mission
4. FA Unit Fires Mission
7. Friendly Unit Updates Ammo On hand
9. Terrain Mgr updates False Targets in Terrain
10. FA Unit notifies FSE of Msn status
EVENTS
2. Friendly FSE receives requests and assigns Msn
5. Enemy unit resolves attrition
6. Attrition Mgr returns Ammo Consumed
8. Attrition Mgr broadcasts False Targets in Terrain
11. FSE receives Msn status
Eagle Simulation
Eagle Simulation
Commo Interaction Event 1
Commo Interaction Event 2
Indirect Fire Interaction Event 4
Ammo Used Interaction Event 6
Terrain DB Interaction Event 8
Commo Interaction Event 10
Network

Figure 10 Eagle Interaction Sequence

Whereas the set of interactions in the Eagle Early Analysis Experiment is an exhaustive set of all possible actions, the set used for the JTFp FOM was derived as part of the negotiations between the federates based on the scenario that was to be played. The interactions can be categorized in two groups: interactions between units and federation management interactions (Figure 11). Of the 23 interaction defined in the JTFp FOM, Eagle subscribed and published to ten.



0 Unit Interactions
- Detection (implicit)
- Indirect Fire
- Air to surface
- Surface to air
- Communications

0 Federation Management Interactions
- Initialization

Interaction	Initiating	Receiving
TBM Warming TBMLaunchAlert Situation Report	Player Player Player - Eagle Commo Manager	Eagle Commo Manager - Player Eagle Commo Manager- Player Player (JTF HQ)
RequestAirSupport	Player - Eagle Commo Manager	Player (JTF HQ)
AirToAggregate GroundEngage	Player	Eagle Attrition Manager
DiscreteGroundToAirEngage AggregateGround ToAirEngage Get & Return LOS	Player Eagle Attrition Manager Eagle Terrain Manager	Eagle Attrition Manager Player Surface Cover
Initialize Federation	Federation Status	Eagle Federation Status
Execute Federation	Federation Status	Eagle Federation Status

JTFp has defined 24 Interactions
Eagle subscribes/published to 10

Figure 11 JTFp Interactions

Just as in the declaration of the attributes associated with the negotiated object classes, parameters associated with each of these interactions had to also be negotiated. The final definition was a compromise between the federates and hence Eagle had to translate between the form used within Eagle and that defined in the JTFp FOM.

The defining of a FOM is a mandatory step in creating a federation. Not only is it required by the HLA compliance rules, but in a practical sense it is the basis upon which each federate proceeds with their own implementation. During the integration testing of the JTFp federation it was not uncommon for inconsistencies to occur. In every case, the base line document that was used to resolve the problem was the JTFp FOM. The FOM is the contract between the simulation participants and its early definition in the evolution of creating a federation can not be over emphasized.

6.0 TIME MANAGEMENT AND EVENT SYNCHRONIZATION

Time management and event synchronization are key areas of concern within the analytical community. Our simulations require that causality between events be maintained so that subsequent analysis of battle outcomes can be validated and verified. Important decisions on the future of the Army are many times based on this analysis; therefore, consistency in the execution of our simulations is of primary importance.

To meet the analytical community's requirements and the associated concerns of the training community, the HLA has provided Time Management services. These time management services provide the means to maintain event synchronization and causality between federates to whatever degree is required by the application. The management of time was a key design element of making Eagle HLA compliant.

6.1 Overview of Eagle's time management.

Eagle is a combat analytical simulation that has a hybrid event structure that relies on both the notion of continuous time and the projecting of discrete events within a limited look ahead time (Figure 12).

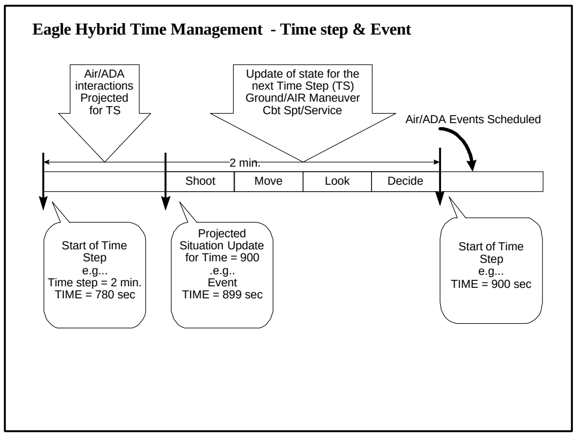

Eagle Hybrid Time Management - Time step & Event
Air/ADA interactions Projected for TS
Update of state for the next Time Step (TS) Ground/AIR Maneuver Cbt Spt/Service
Air/ADA Events Scheduled
2 min.
Shoot
Move
Look
Decide
Start of Time Step e.g... Time step = 2 min. TIME = 780 sec
Projected Situation Update for Time = 900 .e.g.. Event TIME = 899 sec
Start of Time Step e.g... TIME = 900 sec

Figure 12 Eagle Time Management

Continuous time is simulated with a time step scheme that is normally set at one to five minutes. Within this "time step" causality is maintained by the simulation with any discrete events that may occur. Continuous events are executed prior to each time step bringing the simulation "up to state" for the given time. Simulation algorithms that support these events are unique to this notion of the event occurring over a continuous time rather than discrete time. Eagle classifies all ground maneuver actions as continuous. This includes events such as shooting direct and indirect fires, conducting engineer work, moving, consuming, detecting, intelligence fusion, maneuver command & control, coordinating fire support, coordinating engineer support, coordinating air support, and communicating. Eagle classifies all air to ground and ground to air interactions as discrete events. Eagle allows air units to project their path over the time step. Air defense (AD) units determine discrete points of interaction between the maximum range of their weapon system and the projected paths of the air units. Given the unique decision process of the AD units, all air-ground events are determined for the time step. These events are time ordered and then executed. After each event is executed, the air unit is given an opportunity to change its flight. If the unit changes its flight path, all AD units are given an opportunity to reallocate their firings. This may entail the adding and removing of events on the event queue. This process of discrete event air/AD interactions continues until no more events are projected to occur within the time step.

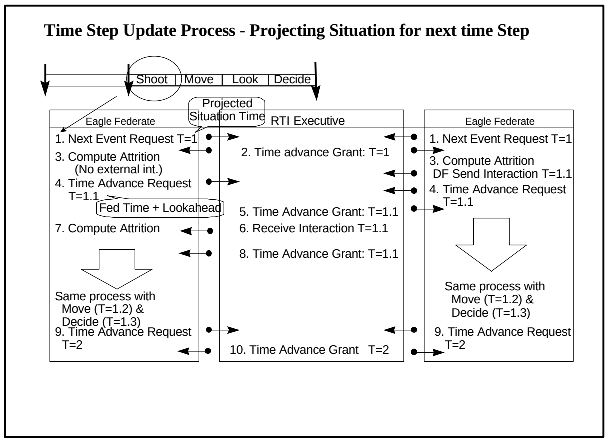
6.2 Eagle and the RTI

The challenge of integrating Eagle with the RTI is to maintain a consistent view of time within Eagle regardless of the types of simulations that are participating in a federation. The Eagle Early Analysis Experiment uses the following time management schemes: timestepped actions, event actions, as-fast-as-possible, and conservative (non-rollback-based) synchronization protocols. Eagle federates have the normal capability to disaggregate combat units into the Distributed Interactive Simulation (DIS) environment. Consistency is required to be maintained between the Eagle "constructive" simulation and the DIS "virtual" simulation. Time management schemes used to maintain this consistency are: timestepped actions, event actions, "independent" time advance, and scaled real-time (constrained). The JTFp uses the following time management schemes: timestepped actions, event actions, scaled real-time (constrained), and optimistic (rollback-based) synchronization protocols

A basic requirement for Eagle to participate in each of these ProtoFederations is that there can not be a unique Eagle for each federation. Therefore, a mixture of event ordering and transportation services is required, allowing Eagle to execute simultaneously in all three ProtoFederations. The basic design decision when incorporating the RTI into Eagle was to subordinate the Eagle's event controller to the RTI's event management schema. Eagle still maintains its event controller, yet when interacting with the RTI, it must receive from the RTI permission to execute each of its events. The event controller must be modified because the RTI may in essence say that an event can not occur because it has an external event for the simulation to consider before it can execute the requested event.

6.3 The Eagle Early Analysis Experiment Time Management

As stated previously, those events that are used to bring an Eagle model "up to state" are classified as continuous events. These events which can be summarized as shoot, move, look, and decide all have the same execution time but are coordinated with a separate numerical prioritization. Within Eagle, the sensitivity of the algorithms that model the physical activities of a unit is affected by the interval of the time step. For example, at a time step of five minutes, the distance moved by a combat unit may exceed the distance intervals used to determine the coefficients of the attrition algorithms. Thus, to be fair, all unit movement must be coordinated and occur before the resolution of attrition. If each unit was allowed to independently move then shoot, an unfair battle would occur because one unit would be closer to the other when it had its chance to shoot. As the time step gets smaller the distance moved is smaller, thus this coordination structure is not necessary (such as in DIS). However, given the relatively large time steps of the Eagle model, this control is needed and the time management services are used to effect this coordination between Eagle federates when executing these continuous events. The basic principle is that Eagle coordinates these events by incrementing time at a very small interval (called the update state interval). Eagle executes events to the nearest second; therefore, a coordinating time of .1 seconds is used to ensure that this update process does not interfere with any normal air AD events.



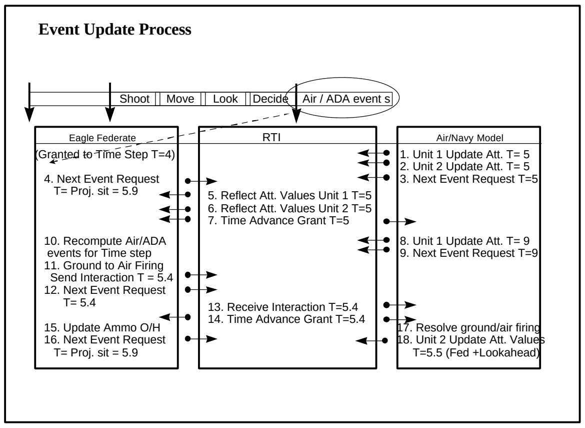
Time Step Update Process - Projecting Situation for next time Step
Shoot
Move
Look
Decide
Projected Situation Time
Eagle Federate
RTI Executive
Eagle Federate
1. Next Event Request T=1
3. Compute Attrition (No external int.)
4. Time Advance Request T=1.1
Fed Time + Lookahead
7. Compute Attrition
Same process with Move (T=1.2) & Decide (T=1.3)
9. Time Advance Request T=2
2. Time advance Grant: T=1
5. Time Advance Grant: T=1.1
6. Receive Interaction T=1.1
8. Time Advance Grant: T=1.1
10. Time Advance Grant T=2
1. Next Event Request T=1
3. Compute Attrition DF Send Interaction T=1.1
4. Time Advance Request T=1.1
Same process with Move (T=1.2) & Decide (T=1.3)
9. Time Advance Request T=2

Figure 13 Eagle Time Step Updates

The example shown in Figure 13, begins with each Eagle federate requesting to go to the projected update situation time. Upon approval from the RTI with a time advance grant, each Eagle federate computes its attrition. During this process, one Eagle federate outputs an attrition interaction. The federation time used on the interaction is the current federation time plus the update state interval which in this case is the same as the lookahead time. Upon finishing computing all internal attrition and outputting all external attrition, each federate requests to advance to its next coordinating time which is always the current federation time plus the update state interval. The federate will receive all object management requests from the RTI with this same time and then receive a time advance grant. This process continues through each of Eagle's update state intervals thus insuring that consistency exists between the multiple data bases prior to the information being needed by the local Eagle service. For example, all units will have moved to their new location and their reflected representations updated prior to the Eagle event of detection.

6.4 The Joint Training ProtoFederation Time Management

Whereas Eagle uses both time step and event time management schemes, the JTFp federates use only an event time management scheme. For Eagle, this meant that as Eagle requested to advance to a desired time, object management requests could be received with a time stamp less then that requested. The RTI would deliver these requests and allow Eagle to advance to a time short of that requested. Figure 14 depicts a typical set of exchanges between Eagle and one of the federates.



Event Update Process
Shoot
Move
Look
Decide
Air / ADA event s
Eagle Federate
RTI
Air/Navy Model
(Granted to Time Step T=4)
4. Next Event Request T= Proj. sit = 5.9
10. Recompute Air/ADA events for Time step
11. Ground to Air Firing Send Interaction T = 5.4
12. Next Event Request T= 5.4
15. Update Ammo O/H
16. Next Event Request T= Proj. sit = 5.9
5. Reflect Att. Values Unit 1 T=5
6. Reflect Att. Values Unit 2 T=5
7. Time Advance Grant T=5
13. Receive Interaction T=5.4
14. Time Advance Grant T=5.4
1. Unit 1 Update Att. T= 5
2. Unit 2 Update Att. T= 5
3. Next Event Request T=5
8. Unit 1 Update Att. T= 9
9. Next Event Request T=9
17. Resolve ground/air firing
18. Unit 2 Update Att. Values T=5.5 (Fed +Lookahead)

Figure 14 Eagle Event Updates

In the example shown in Figure 14, the Eagle federate is requesting to go to the projected update situation time. However, the RTI has object management messages from the Air/Navy model with a time stamp less then that requested. So in this case, the RTI delivers these messages and approves Eagle to advance to this new time short of that requested. Within the Eagle model, depending on the type of object management request, the message will be processed. In this case, as these are reflect attribute value calls, Eagle will allow the Air Defense (AD) to reassess its ability to fire on the planes. The example shows that one of the AD units has decided to fire and an interaction is sent to the RTI. Eagle now no longer wants to go to its projected update state, but to the event time of the Air/AD interaction. The RTI eventually grants Eagle to the Air/AD interaction time, at which time additional internal processing occurs. Finally, Eagle again requests to go to its projected update situation time. If

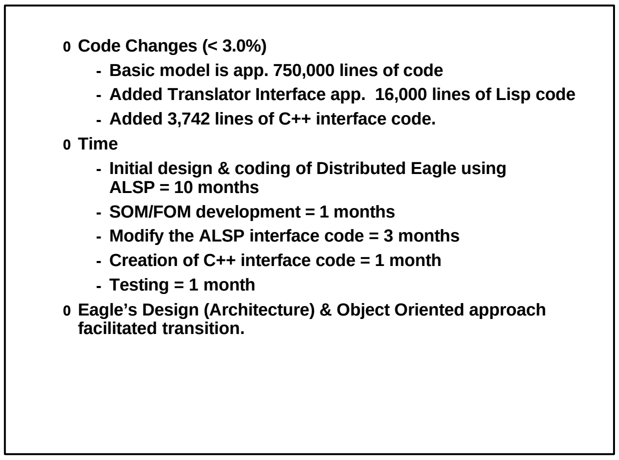
Eagle had received an additional reflect attribute value call prior to the Air/AD interaction event that would have invalidated the Air/AD interaction, Eagle would have used the retract event service provided by the RTI. This exchange demonstrates the use of the time management services to maintain causality between the federates in the execution of the scenario.

The RTI time management services provide a very flexible and robust means to coordinate the events between federates. Eagle's participation in multiple federations each with a different time management scheme demonstrates that the RTI can support the needs of the analytical community.

7.0 SUMMARY

Eagle successfully participated in each of these efforts. Limited testing was accomplished comparing the ALSP distributed version of Eagle with the new HLA distributed version. This testing showed that the HLA version had better performance and, as with the ALSP version, was able to guarantee the delivery of messages so that multiple runs of the model resulted in the same sequence of actions and outcomes. The RTI allows an analytical model to maintain consistency across a distributed network using time and events as a means to synchronize the actions and to maintain a consistent view of the battlefield.

Summary statistics of this effort are shown in Figure 15.



0 **Code Changes (< 3.0%)**
- **Basic model is app. 750,000 lines of code**
- **Added Translator Interface app. 16,000 lines of Lisp code**
- **Added 3,742 lines of C++ interface code.**

0 **Time**
- **Initial design & coding of Distributed Eagle using ALSP = 10 months**
- **SOM/FOM development = 1 months**
- **Modify the ALSP interface code = 3 months**
- **Creation of C++ interface code = 1 month**
- **Testing = 1 month**

0 **Eagle's Design (Architecture) & Object Oriented approach facilitated transition.**

Figure 15 Summary Statistics

The HLA provides an excellent vehicle for the analytical community to meet its current and future needs. Through the Object Model Template process, the HLA will allow the simulation community at large to understand the richness of our analytical simulations. In addition, through the RTI and associated interface specification, the HLA provides an excellent means to share data within a simulation but more importantly to share data with other simulations.

8.0 REFERENCES

Alexander, Robert S., Intelligent Applications of Artificial Intelligence, **The Bulletin of Military Operations Research - PHALANX** (Volume 24, Number 4), December 1991, p 20-23.